What is claimed is:

1. A method of maintaining a conversation between a server and a client using either a bi-directional or a non bi-directional communication protocol, the method comprising:

creating a client software object at the client to initiate a connection with the server and to manage a conversation between the server and the client from the perspective of the client;

selecting a selected communication protocol from a set of available communication protocols that are supported by the client, the server, and any intermediary servers between the client and the server;

if the selected communication protocol is a bi-directional communication protocol, then:

exchanging bi-directional communication protocol messages between the server and the client; and

transferring control over the conversation to the bi-directional communication protocol;

if the selected communication protocol is a non-bi-directional communication protocol, then:

exchanging regular non-bi-directional communication protocol messages between the server and the client;

inserting additional non-bi-directional communication protocol messages as needed to maintain the conversation; and connecting a plurality of the regular and additional non-bi-directional communication protocol messages as the conversation to emulate bi-directional communication between the client and the server; and

layering client application programming over the client software object and the conversation so that the conversation appears fully bi-directional to the client application programming.

2. The method according to claim 1, further comprising:

determining the set of available communication protocols at the client.

3. The method according to claim 2, wherein the act of selecting gives preference to bi-directional communication protocols of the set of available communication protocols.

4. The method according to claim 3, further comprising:

initiating a connection to the server using the selected communication protocol.

5. The method according to claim 4, further comprising:

sending an initial request to the server over the connection.

6. The method according to claim 5, further comprising:

receiving the connection at the server;

detecting the selected communication protocol at the server; and

dispatching the initial request at the server.

7. The method according to claim 6, further comprising:

creating a server software object at the server responsively to the initial request to manage a remainder of the conversation between the server and the client from the perspective of the server.

8. The method according to claim 7, further comprising:

layering server application programming over the server software object and the conversation so that the conversation appears fully bi-directional to the server application programming.

9. A method of maintaining a conversation between a server and a client using as a communication protocol either a supported one of a plurality of bi-directional communication protocols or an available one of a plurality of non bi-directional communication protocols, depending on whether a need exists to communicate via at least one intermediary server that does not support any of the plurality of bi-directional communication protocols, the method comprising:

creating a server software object at the server to process protocol messages and to manage a conversation between the server and the client from the perspective of the server;

if there is no intermediary server present between the server and the client that does not support any of the plurality of bi-directional communication protocols, then:

exchanging supported bi-directional communication protocol messages between the server and the client; and

transferring control over the conversation to the supported bi-directional communication protocol;

if at least one intermediary server is present between the server and the client that does not support any of the plurality of bi-directional communication protocols, then:

exchanging messages of a first type according to the available non-bi-directional communication protocol between the server and the client;

inserting messages of a second type according to the available non-bi-directional communication protocol as needed to maintain the conversation; and

connecting a plurality of the messages of the first type and the messages of the second type as the conversation to emulate bi-directional communication between the client and the server; and

layering application programming over the server software object and the conversation so that the conversation appears fully bi-directional to the application programming and so that the messages of the second type are concealed from the application programming.

10. A method of maintaining a conversation between a server and a client using as a communication protocol either a bi-directional communication protocol or a non bi-directional communication protocol, depending on whether a need exists to communicate via at least one intermediary server that does not support the bi-directional communication protocol, the method comprising:

creating a server software object at the server to process protocol messages and to manage a conversation between the server and the client from the perspective of the server;

if there is no intermediary server present between the server and the client that does not support the bi-directional communication protocol, then:

exchanging bi-directional communication protocol messages between the server and the client; and

transferring control over the conversation to the bi-directional communication protocol;

if at least one intermediary server is present between the server and the client that does not support the bi-directional communication protocol, then:

exchanging regular non-bi-directional communication protocol messages between the server and the client;

inserting additional non-bi-directional communication protocol messages as needed to maintain the conversation; and

connecting a plurality of the regular and additional non-bi-directional communication protocol messages as the conversation to emulate bi-directional communication between the client and the server; and

layering application programming over the server software object and the conversation so that the conversation appears fully bi-directional to the application programming.

11. A method of maintaining a conversation between a server and a client using as a communication protocol either a bi-directional communication protocol or a non bi-directional communication protocol, depending on whether a need exists to communicate via at least one intermediary server that does not support the bi-directional communication protocol, the method comprising:

creating a server software object at the server to process protocol messages and to manage a conversation between the server and the client from the perspective of the server;

determining whether or not there is at least one intermediary server present between the server and the client;

if there is no intermediary server present, then;

exchanging bi-directional communication protocol messages between the server and the client; and

transferring control over the conversation to the bi-directional communication protocol;

if at least one intermediary server is present, the at least one intermediary server being compatible with the non bi-directional communication protocol, then;

exchanging non-bi-directional communication protocol messages between the server and the client;

inserting additional non-bi-directional communication protocol messages as needed to maintain the conversation; and

connecting a plurality of the non-bi-directional communication protocol messages as the conversation to emulate bi-directional communication between the client and the server; and

layering application programming over the server software object and the conversation so that the conversation appears fully bi-directional to the application programming.

12. The method according to claim 11, wherein the additional non-bi-directional communication protocol messages are concealed from the application programming.

13. The method according to claim 11, further comprising:

creating a client software object at the client to process protocol messages and to manage the conversation between the client and the server from the perspective of the client.

14. The method according to claim 13, wherein the server software object consumes the additional non-bi-directional communication protocol messages so that the application programming is isolated from the additional non-bi-directional communication protocol messages.

15. The method according to claim 14, wherein the at least one intermediary server comprises at least one proxy server.

16. The method according to claim 14, wherein the at least one intermediary server comprises at least one relay server.

17. The method according to claim 11, wherein the non-bi-directional communication protocol is limited to a single request and a single response for a duration of the conversation.

18. The method according to claim 11, wherein the non bi-directional communication protocol is HTTP.

19. A method of emulating a bi-directional communications connection between a server and a client, the method comprising:

creating a server software object at the server to control the exchange of communication protocol messages between the server and the client from the perspective of the server and to isolate higher level server programming that calls the server software object from the exchange;

determining whether or not the server and the client are exchanging communication protocol messages via at least one intermediary server;

transferring control over the exchange of communication protocol messages away from the server software object if the server and the client are not exchanging communication protocol messages via at least one intermediary server; and otherwise,

connecting a plurality of the communication protocol messages as a conversation, the plurality of the communication protocol messages each including an identifier that is associated with the server software object and the conversation.

20. The method according to claim 19, further comprising:

creating a client software object at the client to control the exchange of the communication protocol messages between the client and the server from the perspective of the client and to isolate client higher level programming that calls the client software object from the exchange.

21. A method of automatically adapting to the presence of an intermediary server between a server and a client, the method comprising:

receiving an initial request from the client at the server, the initial request intended to begin a conversation between the server and the client;

creating a first server software object to handle the conversation;

associating a first circuit identifier with the first server software object;

sending a response to the client as part of the conversation;

blocking further processing of the conversation until a second request that includes the first circuit identifier is received from the client; and

if the second request received from the client at the server includes the first circuit identifier; then

finding the first server software object;

unblocking processing of the conversation; and

if the second request was passed through an intermediary server, then

sending at least one successive response to the client, wherein each successive response of the at least one successive response includes the first circuit identifier and triggers a corresponding request from the client that includes the first circuit identifier, except for a final response of the at least one successive response.

22. The method according to claim 21, further comprising:

processing the response and the successive responses at the client using a client software object that corresponds to the conversation.

23. A method of automatically adapting to the presence of an intermediary server between a server and a client, the method comprising:

a). receiving a request at the server from the client;

b). if the request does not include a circuit identifier, then

b1). if the request belongs to a first request category, then

b1a). sending a single response to the client; and b2). if the request belongs to a second request category and is intended to begin a first conversation between the server and the client, then

b2a). creating a first server software object to handle the first conversation;

b2b). if no intermediary server is located between the client and the server, then

b2b1). allowing the client and the server to communicate at will over a direct connection; and

b2c). if an intermediary server is located between the client and the server, then

b2c1). creating a first circuit identifier that is associated with the first server software object;

b2c2). sending a response that includes the first circuit identifier to the client via the intermediary server; and

b2c3). successively and in turn, receiving requests from and sending responses to the client, the requests and responses including the first circuit identifier, the requests being processed at the server using the first server software object; and

c). if the request includes a circuit identifier, then

c1). finding the server software object that is associated with the circuit identifier; and

c2). processing the request at the server using the server software object, the server software object corresponding to a conversation between the server and the client.

24. The method according to claim 23, further comprising:

terminating the first server software object upon completion of the first conversation.

25. The method according to claim 23, wherein the second request category implies a more complex first conversation between the server and the client than the first request category.

26. A method of processing requests at a server, the server in communication with at least one client; the method comprising:

flexibly routing and processing requests from a client according to whether or not the request is intended to open one conversation between the server and the client, whether or not the request includes a circuit identifier that corresponds to a previously existing other conversation between the server and the client, and whether or not the request arrived from the client via an intermediary server connected between the server and the client; and

isolating higher level programming from the routing and processing of the requests.

27. An article, comprising:

a computer-readable medium having instructions stored thereon which process requests at a server, the server in communication with at least one client, which, when the instructions are executed, cause requests from a client to be flexibly routed and processed according to whether or not the request is intended to open one conversation between the server and the client, whether or not the request includes a circuit identifier that corresponds to a previously existing other conversation between the server and the client, and whether or not the request arrived from the client via an intermediary server connected between the server and the client, and cause higher level programming to be isolated from the routing and processing of the requests.

28. A server to communicate with a client, comprising:

a software object to determine whether or not a client is using any intermediary server that is compatible with a non-bi-directional communication protocol to communicate with the server, and to flexibly manage a conversation between the server and the client according to a bi-directional communication protocol if the any intermediary server is not being used and according to the non-bi-directional communication protocol if the any intermediary server is being used;

a request processing interface to receive requests from the client and to create the software object responsively to receiving any request from the client that belongs to a first request category; and

a software routine to call the software object to perform a task that is associated with the first request category and to run an arbitrary communication protocol with no awareness of whether or not the client is using any intermediary server to communicate with the server, the software routine operating at a higher level than the software object.

29. A system that automatically adapts communications between a server and a client to the presence or absence of an intermediary server between the server and the client and isolates a higher-level application program from the details of the communications, comprising:

a server to create a server software object, to receive requests, and to send responses;

a client to select which protocol of at least two communication protocols to use to communicate with the server based on a destination address, to create a client software object, to send requests and to receive responses, wherein the server software object and the client software object allow an application program to communicate using the at least two communication protocols without direct knowledge of the at least two communication protocols, one protocol of the at least two communication protocols being bi-directional to allow the client to send requests to the server and the server to send responses to the client once a connection between the server and the client has been initiated, and an other protocol of the at least two communication protocols being non-bi-directional to allow only a single request from the client and a single response from the server per connection that is initiated between the server and the client, and wherein the client selects the other protocol to use if the destination address designates an intermediary server, and wherein the server sends a response to the client according to a type of request that is sent by the client, and wherein the server, when responding to a first type of request from the client that requires a single response, sends a response according to whichever protocol of the at least two communication protocols the first type of request corresponds, and wherein the server, when responding to a second type of request from the client, the second type of request requiring a more complex conversation with the client, uses the server software object to manage the conversation, and wherein if the client uses the other protocol to communicate with the server, the server assigns a circuit identifier to the software object and routes future requests carrying the circuit identifier to the software object, and wherein the client and the server insert additional messages as needed to maintain the conversation when the other protocol is used.

30. A client to initiate a connection to, to communicate with, and to send a request to a server in a communications network, the client selecting which protocol of at least two communication protocols to use to communicate with the server based on a destination address, the client comprising:

a software object to flexibly manage a conversation between the server and the client according to one protocol of the at least two communication protocols if the client uses an intermediary server that is compatible with a non-bi-directional communication protocol to send the request to the server and according to an other protocol of the at least two communication protocols if an intermediary server is not used to send the request, wherein the client creates the software object upon initiating the connection to the server, and wherein if the client used the intermediary server to send the request to the server, the client receives a circuit identifier in a response from the server and the client includes the circuit identifier in subsequent requests that the client sends to the server, the circuit identifier being used by the server to identify and route requests from the client, and wherein the client inserts additional requests in communication with the server to maintain the conversation if the client is managing the conversation according to the one protocol.